1100 XP

Fundamental AI Concepts

36 min • Module • 10 Units

Feedback

Machine learning in Microsoft Azure

Microsoft Azure provides the **Azure Machine Learning** service - a cloud-based platform for creating, managing, and publishing machine learning models. **Azure Machine Learning Studio** offers multiple authoring experiences such as:

- **Automated machine learning**: this feature enables non-experts to quickly create an effective machine learning model from data.
- **Azure Machine Learning designer**: a graphical interface enabling no-code development of machine learning solutions.
- **Data metric visualization**: analyze and optimize your experiments with visualization.
- **Notebooks**: write and run your own code in managed Jupyter Notebook servers that are directly integrated in the studio.

Understand computer vision

100 XP

5 minutes

Computer Vision is an area of AI that deals with visual processing. Let's explore some of the possibilities that computer vision brings.

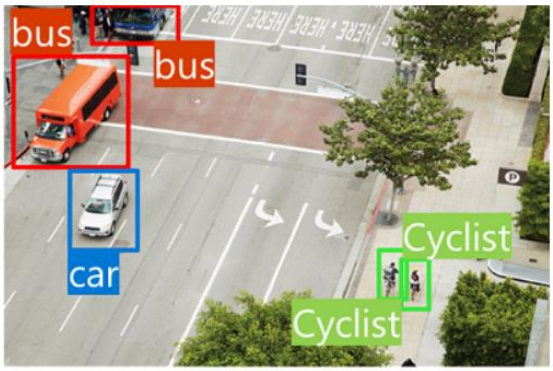
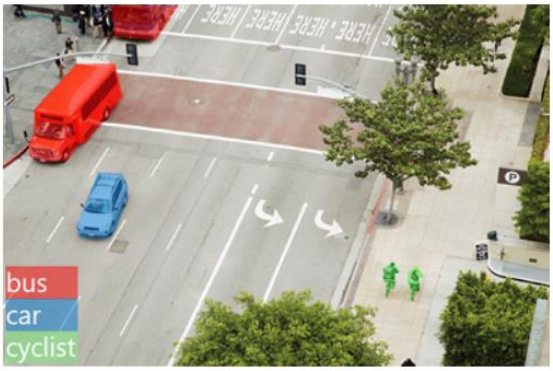
The **Seeing AI** app is a great example of the power of computer vision. Designed for the blind and low vision community, the Seeing AI app harnesses the power of AI to open up the visual world and describe nearby people, text and objects.

Computer Vision models and capabilities

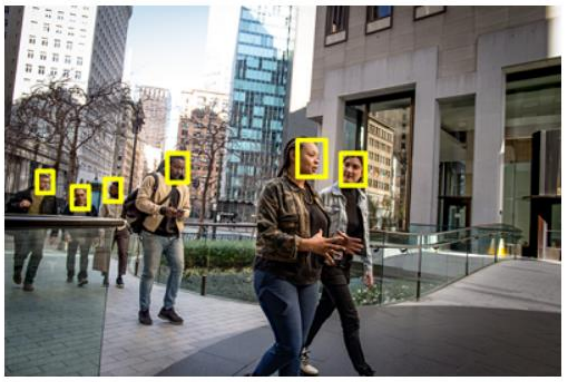
Most computer vision solutions are based on machine learning models that can be applied to visual input from cameras, videos, or images. The following table describes common computer vision tasks.

Expand table

Task	Description
Image classification	 Image classification involves training a machine learning model to classify images based on their contents. For example, in a traffic monitoring solution you might use an image classification model to classify images based on the type of vehicle they contain, such as taxis, buses, cyclists, and so on.

Object detection	 Object detection machine learning models are trained to classify individual objects within an image, and identify their location with a bounding box. For example, a traffic monitoring solution might use object detection to identify the location of different classes of vehicle.
Semantic segmentation	 Semantic segmentation is an advanced machine learning technique in which individual pixels in the image are classified according to the object to which they belong. For example, a traffic monitoring solution might overlay traffic images with "mask" layers to highlight different vehicles using specific colors.
Image analysis	 You can create solutions that combine machine learning models with advanced image analysis techniques to extract information from images, including "tags" that could help catalog the image or even descriptive captions that summarize the scene shown in the image.

Face detection, analysis, and recognition



Face detection is a specialized form of object detection that locates human faces in an image. This can be combined with classification and facial geometry analysis techniques to recognize individuals based on their facial features.

Optical character recognition (OCR)

Optical character recognition is a technique used to detect and read text in images. You can use OCR to read text in photographs (for example, road signs or store fronts) or to extract information from scanned documents such as letters, invoices, or forms.

Computer vision services in Microsoft Azure

You can use Microsoft's **Azure AI Vision** to develop computer vision solutions. The service features are available for use and testing in the **Azure Vision Studio** and other programming languages. Some features of Azure AI Vision include:

- **Image Analysis**: capabilities for analyzing images and video, and extracting descriptions, tags, objects, and text.
- **Face**: capabilities that enable you to build face detection and facial recognition solutions.
- **Optical Character Recognition (OCR)**: capabilities for extracting printed or handwritten text from images, enabling access to a digital version of the scanned text.

Understand natural language processing

100 XP

4 minutes

Natural language processing (NLP) is the area of AI that deals with creating software that understands written and spoken language.

NLP enables you to create software that can:

- Analyze and interpret text in documents, email messages, and other sources.
- Interpret spoken language, and synthesize speech responses.
- Automatically translate spoken or written phrases between languages.
- Interpret commands and determine appropriate actions.

For example, *Starship Commander* is a virtual reality (VR) game from Human Interact that takes place in a science fiction world. The game uses natural language processing to enable players to control the narrative and interact with in-game characters and starship systems.

Natural language processing in Microsoft Azure

You can use Microsoft's **Azure AI Language** to build natural language processing solutions. Some features of Azure AI Language include understanding and analyzing text, training conversational language models that can understand spoken or text-based commands, and building intelligent applications.

Microsoft's **Azure AI Speech** is another service that can be used to build natural language processing solutions. Azure AI Speech features include speech recognition and synthesis, real-time translations, conversation transcriptions, and more.

You can explore Azure AI Language features in the **Azure Language Studio** and Azure AI Speech features in the **Azure Speech Studio**. The service features are available for use and testing in the studios and other programming languages.

Understand document intelligence and knowledge mining

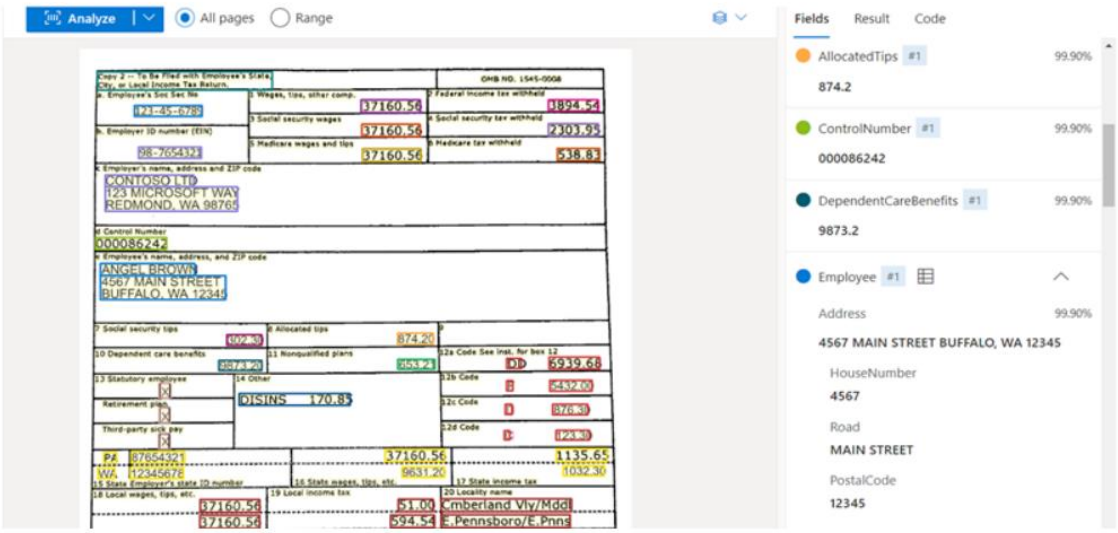
100 XP

3 minutes

Document Intelligence is the area of AI that deals with managing, processing, and using high volumes of a variety of data found in forms and documents. Document intelligence enables you to create software that can automate processing for contracts, health documents, financial forms and more

Document intelligence in Microsoft Azure

You can use Microsoft's **Azure AI Document Intelligence** to build solutions that manage and accelerate data collection from scanned documents. Features of Azure AI Document Intelligence help automate document processing in applications and workflows, enhance data-driven strategies, and enrich document search capabilities. You can use prebuilt models to add intelligent document processing for invoices, receipts, health insurance cards, tax forms, and more. You can also use Azure AI Document Intelligence to create custom models with your own labeled datasets. The service features are available for use and testing in the **Document Intelligence Studio** and other programming languages.



This example shows information extracted from a tax form using Azure AI Document Intelligence.

Knowledge Mining

Knowledge mining is the term used to describe solutions that involve extracting information from large volumes of often unstructured data to create a searchable knowledge store.

Knowledge mining in Microsoft Azure

One Microsoft knowledge mining solution is **Azure AI Search**, an enterprise, search solution that has tools for building indexes. The indexes can then be used for internal only use, or to enable searchable content on public facing internet assets.

Azure AI Search can utilize the built-in AI capabilities of Azure AI services such as image processing, document intelligence, and natural language processing to extract data. The product's AI capabilities makes it possible to index previously unsearchable documents and to extract and surface insights from large amounts of data quickly.

Howden: How they built a knowledge mining solution with Azure Search

By Jeremy Chapman, Director, Microsoft Mechanics

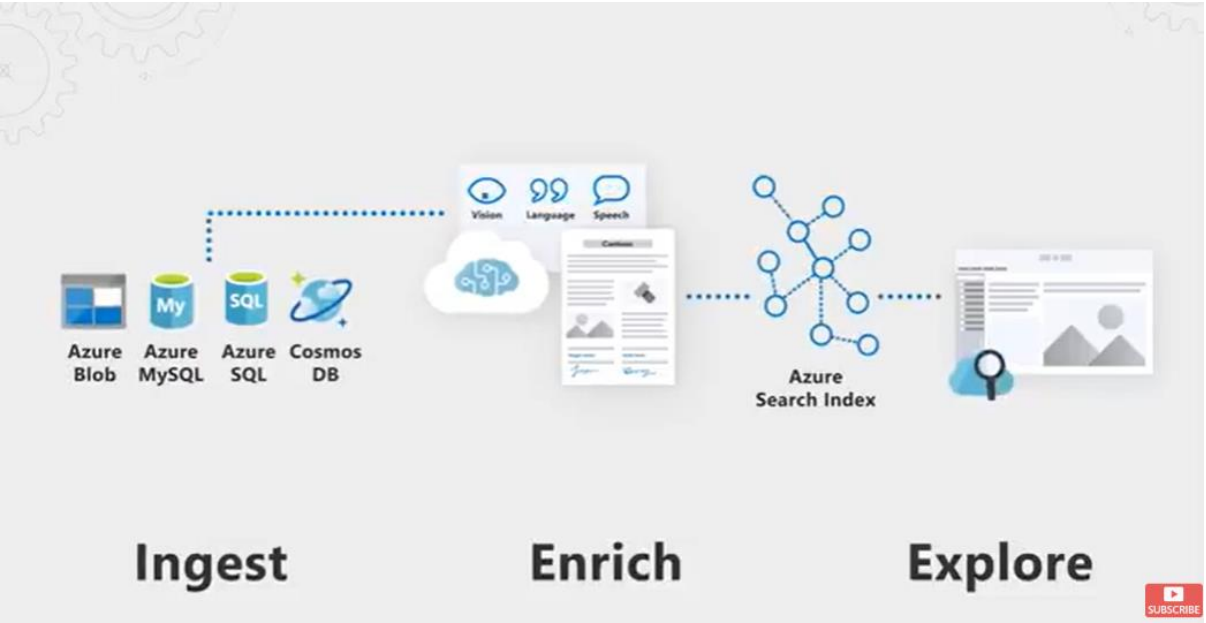
Customers across industries including healthcare, legal, media, and manufacturing are looking for new solutions to solve business challenges with AI, including knowledge mining with Azure Search.

Azure Search enables developers to quickly apply AI across their content to unlock untapped information. Custom or prebuilt cognitive skills like facial recognition, key phrase extraction, and sentiment analysis can be applied to content using the cognitive search capability to extract knowledge that's then organized within a search index. Let's take a closer look at how one company, Howden, applies the cognitive search capability to reduce time and risk to their business.

Howden, a global engineering company, focuses on providing quality solutions for air and gas handling. With over a century of engineering experience, Howden creates industrial products that help multiple sectors improve their everyday processes; from mine ventilation and waste water treatment to heating and cooling.

Too many details, not enough time

Every new project requires the creation of a bid proposal. A typical customer bid can span thousands of pages in differing formats such as Word and PDF. The team has to scour through detailed customer requirements to identify key areas of design and specialized components in order to produce accurate bids. If they miss key or critical details, they can bid too low and lose money, or bid too high and lose the customer opportunity. The manual process is time consuming, labor intensive, and creates multiple opportunities for human error. To learn more about knowledge mining with Azure Search and see how Howden built their solution, check out the Microsoft Mechanics show linked below.



Understand generative AI

100 XP

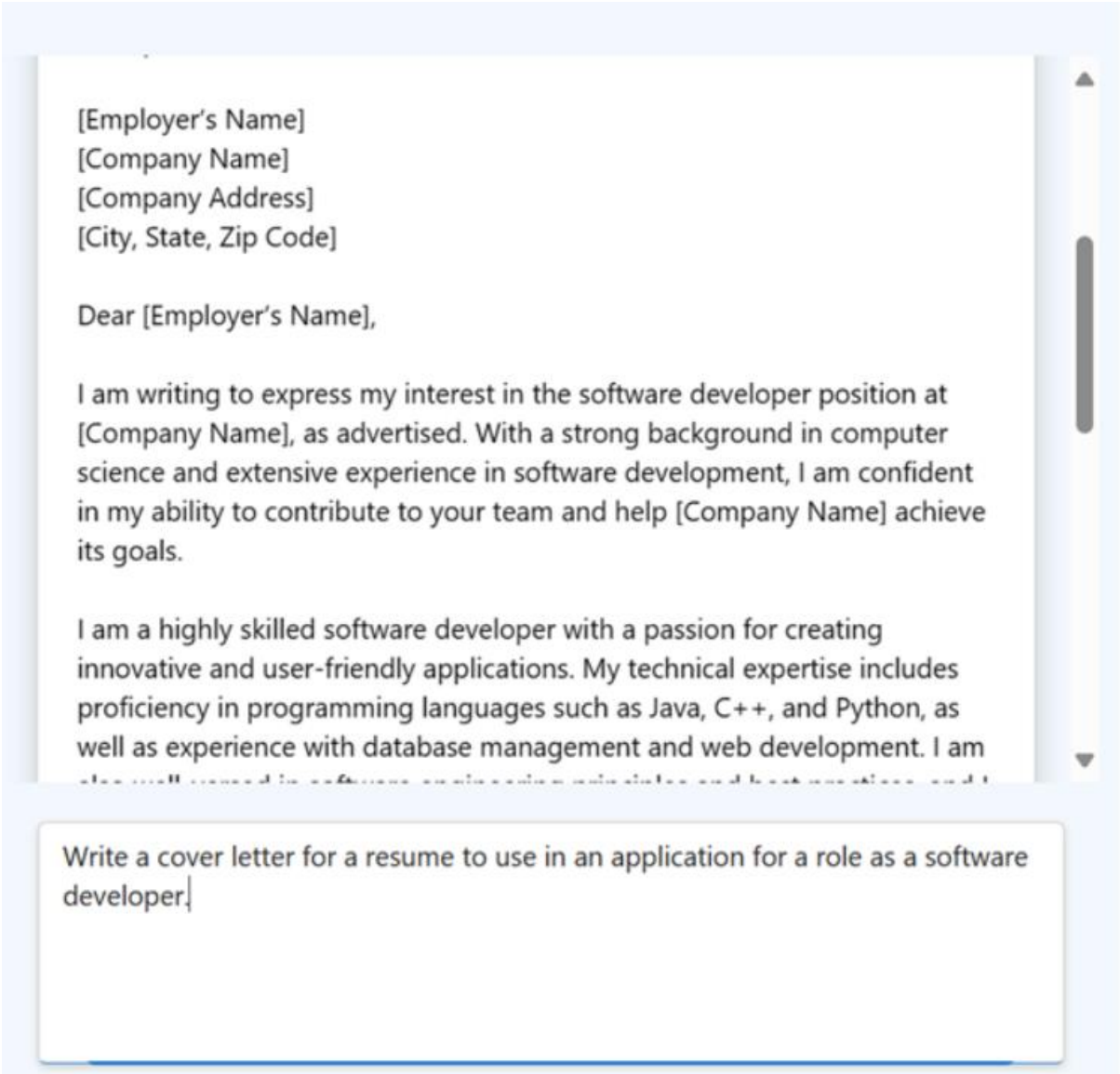
2 minutes

Generative AI describes a category of capabilities within AI that create original content. People typically interact with generative AI that has been built into chat applications. Generative AI applications take in natural language input, and return appropriate responses in a variety of formats including natural language, image, code, and audio.

Generative AI in Microsoft Azure

Azure OpenAI Service is Microsoft's cloud solution for deploying, customizing, and hosting generative AI models. It brings together the best of OpenAI's cutting edge models and APIs with the security and scalability of the Azure cloud platform.

Azure OpenAI Service supports many generative model choices that can serve different needs. You can use **Azure AI Studio** to create generative AI solutions, such as custom *copilot* chat-based assistants that use Azure OpenAI Service models



Challenges and risks with AI

100 XP

3 minutes

Artificial Intelligence is a powerful tool that can be used to greatly benefit the world. However, like any tool, it must be used responsibly.

The following table shows some of the potential challenges and risks facing an AI application developer.

Challenge or Risk	Example
Bias can affect results	A loan-approval model discriminates by gender due to bias in the data with which it was trained
Errors may cause harm	An autonomous vehicle experiences a system failure and causes a collision
Data could be exposed	A medical diagnostic bot is trained using sensitive patient data, which is stored insecurely
Solutions may not work for everyone	A home automation assistant provides no audio output for visually impaired users
Users must trust a complex system	An AI-based financial tool makes investment recommendations - what are they based on?
Who's liable for AI-driven decisions?	An innocent person is convicted of a crime based on evidence from facial recognition – who's responsible?

Understand Responsible AI

100 XP

10 minutes

At Microsoft, AI software development is guided by a set of six principles, designed to ensure that AI applications provide amazing solutions to difficult problems without any unintended negative consequences.

Fairness

AI systems should treat all people fairly. For example, suppose you create a machine learning model to support a loan approval application for a bank. The model should predict whether the loan should be approved or denied without bias. This bias could be based on gender, ethnicity, or other factors that result in an unfair advantage or disadvantage to specific groups of applicants.

Azure Machine Learning includes the capability to interpret models and quantify the extent to which each feature of the data influences the model's prediction. This capability helps data scientists and developers identify and mitigate bias in the model.

Another example is Microsoft's implementation of Responsible AI with the Face service, which retires facial recognition capabilities that can be used to try to infer emotional states and identity attributes. These capabilities, if misused, can subject people to stereotyping, discrimination or unfair denial of services.

Reliability and safety

AI systems should perform reliably and safely. For example, consider an AI-based software system for an autonomous vehicle; or a machine learning model that diagnoses patient symptoms and recommends prescriptions. Unreliability in these kinds of systems can result in substantial risk to human life.

AI-based software application development must be subjected to rigorous testing and deployment management processes to ensure that they work as expected before release.

Privacy and security

AI systems should be secure and respect privacy. The machine learning models on which AI systems are based rely on large volumes of data, which may contain personal details that must be kept private. Even after the models are trained and the system is in production, privacy and security need to be considered. As the system uses new data to make predictions or take action, both the data and decisions made from the data may be subject to privacy or security concerns.

Inclusiveness

AI systems should empower everyone and engage people. AI should bring benefits to all parts of society, regardless of physical ability, gender, sexual orientation, ethnicity, or other factors.

Transparency

AI systems should be understandable. Users should be made fully aware of the purpose of the system, how it works, and what limitations may be expected.

Accountability

People should be accountable for AI systems. Designers and developers of AI-based solutions should work within a framework of governance and organizational principles that ensure the solution meets ethical and legal standards that are clearly defined.

1300 XP

Fundamentals of machine learning

1 hr 54 min • Module • 12 Units

Feedback

Beginner | AI Engineer | Data Scientist | Azure Machine Learning

Machine learning is the basis for most modern artificial intelligence solutions. A familiarity with the core concepts on which machine learning is based is an important foundation for understanding AI.

Learning objectives

After completing this module, you will be able to:

- Describe core concepts of machine learning
- Identify different types of machine learning
- Describe considerations for training and evaluating machine learning models
- Describe core concepts of deep learning
- Use automated machine learning in Azure Machine Learning service

Introduction

100 XP

1 minute

Machine learning is in many ways the intersection of two disciplines - data science and software engineering. The goal of machine learning is to use data to create a predictive model that can be incorporated into a software application or service. To achieve this goal requires collaboration between data scientists who explore and prepare the data before using it to *train* a machine learning model, and software developers who integrate the models into applications where they're used to predict new data values (a process known as *inferencing*).

In this module, you'll explore some of the core concepts on which machine learning is based, learn how to identify different kinds of machine learning models, and examine the ways in which machine learning models are trained and evaluated. Finally, you'll learn how to use Microsoft Azure Machine Learning to train and deploy a machine learning model, without needing to write any code.

> **Note**
>
> Machine learning is based on mathematical and statistical techniques, some of which are described at a high level in this module. Don't worry if you're not a mathematical expert though! The goal of the module is to help you gain an *intuition* of how machine learning works - we'll keep the mathematics to the minimum required to understand the core concepts.

What is machine learning?

100 XP

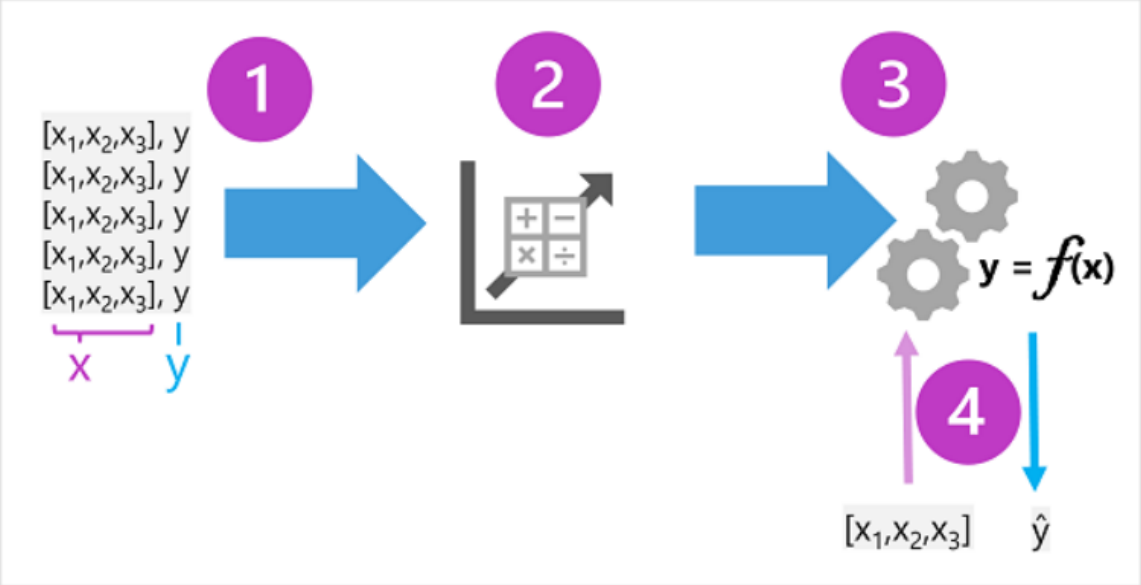
5 minutes

Machine learning has its origins in statistics and mathematical modeling of data. The fundamental idea of machine learning is to use data from past observations to predict unknown outcomes or values. For example:

- The proprietor of an ice cream store might use an app that combines historical sales and weather records to predict how many ice creams they're likely to sell on a given day, based on the weather forecast.
- A doctor might use clinical data from past patients to run automated tests that predict whether a new patient is at risk from diabetes based on factors like weight, blood glucose level, and other measurements.
- A researcher in the Antarctic might use past observations automate the identification of different penguin species (such as *Adelie*, *Gentoo*, or *Chinstrap*) based on measurements of a bird's flippers, bill, and other physical attributes.

Machine learning as a *function*

Because machine learning is based on mathematics and statistics, it's common to think about machine learning models in mathematical terms. Fundamentally, a machine learning model is a software application that encapsulates a *function* to calculate an output value based on one or more input values. The process of defining that function is known as *training*. After the function has been defined, you can use it to predict new values in a process called *inferencing*.

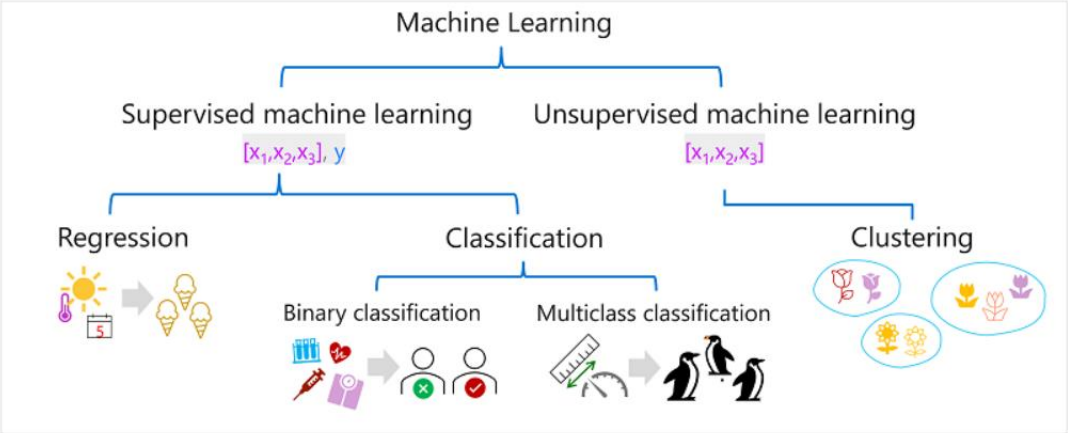


Types of machine learning

100 XP

10 minutes

There are multiple types of machine learning, and you must apply the appropriate type depending on what you're trying to predict. A breakdown of common types of machine learning is shown in the following diagram.



Supervised machine learning

Supervised machine learning is a general term for machine learning algorithms in which the training data includes both *feature* values and known *label* values. Supervised machine learning is used to train models by determining a relationship between the features and labels in past observations, so that unknown labels can be predicted for features in future cases

Regression

Regression is a form of supervised machine learning in which the label predicted by the model is a numeric value. For example:

- The number of ice creams sold on a given day, based on the temperature, rainfall, and windspeed.
- The selling price of a property based on its size in square feet, the number of bedrooms it contains, and socio-economic metrics for its location.
- The fuel efficiency (in miles-per-gallon) of a car based on its engine size, weight, width, height, and length.

Classification

Classification is a form of supervised machine learning in which the label represents a categorization, or *class*. There are two common classification scenarios.

Binary classification

In *binary classification*, the label determines whether the observed item *is* (or *isn't*) an instance of a specific class. Or put another way, binary classification models predict one of two mutually exclusive outcomes. For example:

- Whether a patient is at risk for diabetes based on clinical metrics like weight, age, blood glucose level, and so on.
- Whether a bank customer will default on a loan based on income, credit history, age, and other factors.
- Whether a mailing list customer will respond positively to a marketing offer based on demographic attributes and past purchases.

In all of these examples, the model predicts a binary *true/false* or *positive/negative* prediction for a single possible class.

Multiclass classification

Multiclass classification extends binary classification to predict a label that represents one of multiple possible classes. For example,

- The species of a penguin (*Adelie*, *Gentoo*, or *Chinstrap*) based on its physical measurements.
- The genre of a movie (*comedy*, *horror*, *romance*, *adventure*, or *science fiction*) based on its cast, director, and budget.

In most scenarios that involve a known set of multiple classes, multiclass classification is used to predict mutually exclusive labels. For example, a penguin can't be both a *Gentoo* and an *Adelie*. However, there are also some algorithms that you can use to train *multilabel* classification models, in which there may be more than one valid label for a single observation. For example, a movie could potentially be categorized as both *science fiction* and *comedy*.

Unsupervised machine learning

Unsupervised machine learning involves training models using data that consists only of *feature* values without any known labels. Unsupervised machine learning algorithms determine relationships between the features of the observations in the training data.

Clustering

The most common form of unsupervised machine learning is *clustering*. A clustering algorithm identifies similarities between observations based on their features, and groups them into discrete clusters. For example:

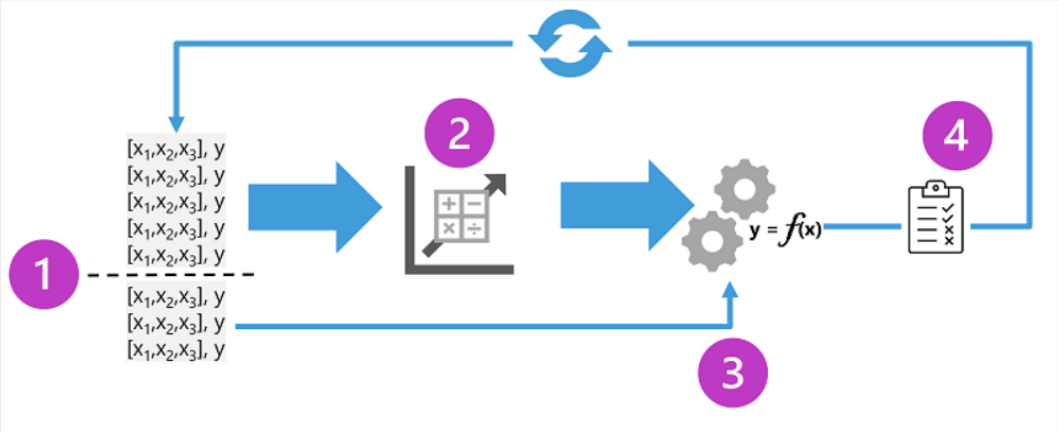
- Group similar flowers based on their size, number of leaves, and number of petals.
- Identify groups of similar customers based on demographic attributes and purchasing behavior.

Regression

100 XP

12 minutes

Regression models are trained to predict numeric label values based on training data that includes both features and known labels. The process for training a regression model (or indeed, any supervised machine learning model) involves multiple iterations in which you use an appropriate algorithm (usually with some parameterized settings) to train a model, evaluate the model's predictive performance, and refine the model by repeating the training process with different algorithms and parameters until you achieve an acceptable level of predictive accuracy.



Regression evaluation metrics

Based on the differences between the predicted and actual values, you can calculate some common metrics that are used to evaluate a regression model.

Mean Absolute Error (MAE)

The variance in this example indicates by how many ice creams each prediction was wrong. It doesn't matter if the prediction was *over* or *under* the actual value (so for example, -3 and +3 both indicate a variance of 3). This metric is known as the *absolute error* for each prediction, and can be summarized for the whole validation set as the **mean absolute error** (MAE).

In the ice cream example, the mean (average) of the absolute errors (2, 3, 3, 1, 2, and 3) is **2.33**.

Mean Squared Error (MSE)

The mean absolute error metric takes all discrepancies between predicted and actual labels into account equally. However, it may be more desirable to have a model that is consistently wrong by a small amount than one that makes fewer, but larger errors. One way to produce a metric that "amplifies" larger errors by *squaring* the individual errors and calculating the mean of the squared values. This metric is known as the **mean squared error** (MSE).

In our ice cream example, the mean of the squared absolute values (which are 4, 9, 9, 1, 4, and 9) is **6**.

Root Mean Squared Error (RMSE)

The mean squared error helps take the magnitude of errors into account, but because it *squares* the error values, the resulting metric no longer represents the quantity measured by the label. In other words, we can say that the MSE of our model is 6, but that doesn't measure its accuracy in terms of the number of ice creams that were mispredicted; 6 is just a numeric score that indicates the level of error in the validation predictions.

If we want to measure the error in terms of the number of ice creams, we need to calculate the *square root* of the MSE; which produces a metric called, unsurprisingly, **Root Mean Squared Error**. In this case √6, which is **2.45** (ice creams).

Coefficient of determination (R^2)

All of the metrics so far compare the discrepancy between the predicted and actual values in order to evaluate the model. However, in reality, there's some natural random variance in the daily sales of ice cream that the model takes into account. In a linear regression model, the training algorithm fits a straight line that minimizes the mean variance between the function and the known label values. The **coefficient of determination** (more commonly referred to as R^2 or **R-Squared**) is a metric that measures the proportion of variance in the validation results that can be explained by the model, as opposed to some anomalous aspect of the validation data (for example, a day with a highly unusual number of ice creams sales because of a local festival).

The calculation for R^2 is more complex than for the previous metrics. It compares the sum of squared differences between predicted and actual labels with the sum of squared differences between the actual label values and the mean of actual label values, like this:

$$R^2 = 1 - \sum(y-\hat{y})^2 \div \sum(y-\bar{y})^2$$

Iterative training

The metrics described above are commonly used to evaluate a regression model. In most real-world scenarios, a data scientist will use an iterative process to repeatedly train and evaluate a model, varying:

- Feature selection and preparation (choosing which features to include in the model, and calculations applied to them to help ensure a better fit).
- Algorithm selection (We explored linear regression in the previous example, but there are many other regression algorithms)
- Algorithm parameters (numeric settings to control algorithm behavior, more accurately called *hyperparameters* to differentiate them from the ***x*** and ***y*** parameters).

After multiple iterations, the model that results in the best evaluation metric that's acceptable for the specific scenario is selected.

Binary classification

100 XP

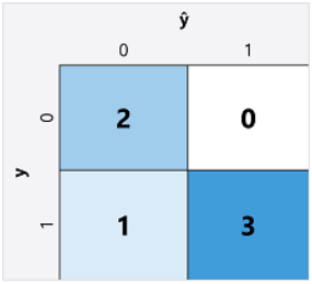
12 minutes

Classification, like regression, is a *supervised* machine learning technique; and therefore follows the same iterative process of training, validating, and evaluating models. Instead of calculating numeric values like a regression model, the algorithms used to train classification models calculate *probability* values for class assignment and the evaluation metrics used to assess model performance compare the predicted classes to the actual classes.

Binary classification algorithms are used to train a model that predicts one of two possible labels for a single class. Essentially, predicting ***true*** or ***false***. In most real scenarios, the data observations used to train and validate the model consist of multiple feature (***x***) values and a ***y*** value that is either **1** or **0**.

Binary classification evaluation metrics

The first step in calculating evaluation metrics for a binary classification model is usually to create a matrix of the number of correct and incorrect predictions for each possible class label:



This visualization is called a *confusion matrix*, and it shows the prediction totals where:

- ŷ=0 and y=0: *True negatives* (TN)
- ŷ=1 and y=0: *False positives* (FP)
- ŷ=0 and y=1: *False negatives* (FN)
- ŷ=1 and y=1: *True positives* (TP)

The arrangement of the confusion matrix is such that correct (*true*) predictions are shown in a diagonal line from top-left to bottom-right. Often, color-intensity is used to indicate the number of predictions in each cell, so a quick glance at a model that predicts well should reveal a deeply shaded diagonal trend.

Accuracy

The simplest metric you can calculate from the confusion matrix is *accuracy* - the proportion of predictions that the model got right. Accuracy is calculated as:

$$(TN+TP) \div (TN+FN+FP+TP)$$

In the case of our diabetes example, the calculation is:

(2+3) ÷ (2+1+0+3)

= 5 ÷ 6

= **0.83**

So for our validation data, the diabetes classification model produced correct predictions 83% of the time.

Accuracy might initially seem like a good metric to evaluate a model, but consider this. Suppose 11% of the population has diabetes. You could create a model that always predicts **0**, and it would achieve an accuracy of 89%, even though it makes no real attempt to differentiate between patients by evaluating their features. What we really need is a deeper understanding of how the model performs at predicting **1** for positive cases and **0** for negative cases.

Recall

Recall is a metric that measures the proportion of positive cases that the model identified correctly. In other words, compared to the number of patients who *have* diabetes, how many did the model *predict* to have diabetes?

The formula for recall is:

$$TP \div (TP+FN)$$

For our diabetes example:

3 ÷ (3+1)

= 3 ÷ 4

= **0.75**

So our model correctly identified 75% of patients who have diabetes as having diabetes.

Precision

Precision is a similar metric to recall, but measures the proportion of predicted positive cases where the true label is actually positive. In other words, what proportion of the patients *predicted* by the model to have diabetes actually *have* diabetes?

The formula for precision is:

TP ÷ (TP+FP)

For our diabetes example:

3 ÷ (3+0)

= 3 ÷ 3

= **1.0**

So 100% of the patients predicted by our model to have diabetes do in fact have diabetes.

F1-score

F1-score is an overall metric that combined recall and precision. The formula for F1-score is:

(2 x Precision x Recall) ÷ (Precision + Recall)

For our diabetes example:

(2 x 1.0 x 0.75) ÷ (1.0 + 0.75)

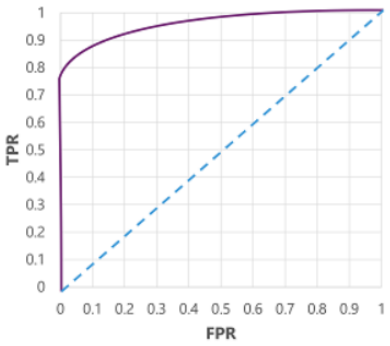
= 1.5 ÷ 1.75

= **0.86**

Area Under the Curve (AUC)

Another name for recall is the *true positive rate* (TPR), and there's an equivalent metric called the *false positive rate* (FPR) that is calculated as **FP÷(FP+TN)**. We already know that the TPR for our model when using a threshold of 0.5 is 0.75, and we can use the formula for FPR to calculate a value of 0÷2 = 0.

Of course, if we were to change the threshold above which the model predicts *true* (**1**), it would affect the number of positive and negative predictions; and therefore change the TPR and FPR metrics. These metrics are often used to evaluate a model by plotting a *received operator characteristic* (ROC) curve that compares the TPR and FPR for every possible threshold value between 0.0 and 1.0:



The ROC curve for a perfect model would go straight up the TPR axis on the left and then across the FPR axis at the top. Since the plot area for the curve measures 1x1, the area under this perfect curve would be 1.0 (meaning that the model is correct 100% of the time). In contrast, a diagonal line from the bottom-left to the top-right represents the results that would be achieved by randomly guessing a binary label; producing an area under the curve of 0.5. In other words, given two possible class labels, you could reasonably expect to guess correctly 50% of the time.

In the case of our diabetes model, the curve above is produced, and the **area under the curve** (AUC) metric is 0.875. Since the AUC is higher than 0.5, we can conclude the model performs better at predicting whether or not a patient has diabetes than randomly guessing.

Multiclass classification

100 XP

12 minutes

Multiclass classification is used to predict to which of multiple possible classes an observation belongs. As a supervised machine learning technique, it follows the same iterative *train, validate, and evaluate* process as regression and binary classification in which a subset of the training data is held back to validate the trained model.

Example - multiclass classification

Multiclass classification algorithms are used to calculate probability values for multiple class labels, enabling a model to predict the *most probable* class for a given observation.

Let's explore an example in which we have some observations of penguins, in which the flipper length (***x***) of each penguin is recorded. For each observation, the data includes the penguin species (***y***), which is encoded as follows:

- 0: Adelie
- 1: Gentoo
- 2: Chinstrap

Clustering

100 XP

10 minutes

Clustering is a form of unsupervised machine learning in which observations are grouped into clusters based on similarities in their data values, or features. This kind of machine learning is considered unsupervised because it doesn't make use of previously known label values to train a model. In a clustering model, the label is the cluster to which the observation is assigned, based only on its features.

Example - clustering

For example, suppose a botanist observes a sample of flowers and records the number of leaves and petals on each flower:

There are no known *labels* in the dataset, just two *features*. The goal is not to identify the different types (species) of flower; just to group similar flowers together based on the number of leaves and petals.

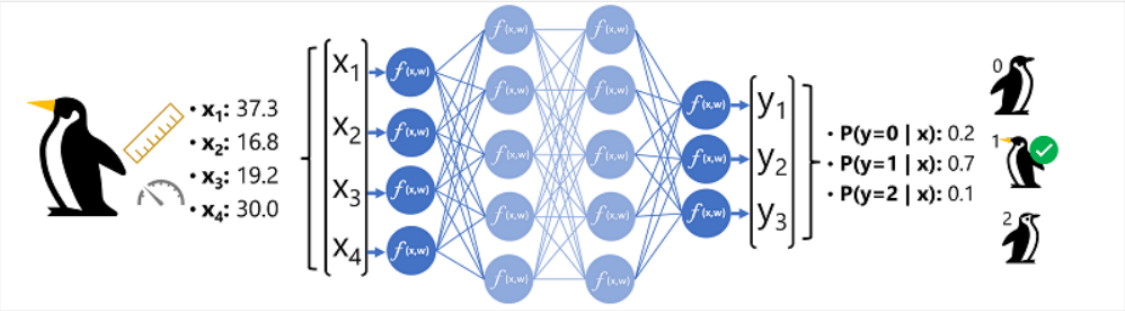
Deep learning

100 XP

12 minutes

Deep learning is an advanced form of machine learning that tries to emulate the way the human brain learns. The key to deep learning is the creation of an artificial *neural network* that simulates electrochemical activity in biological neurons by using mathematical functions, as shown here.

Biological neural network	**Artificial neural network**
	
Neurons fire in response to electrochemical stimuli. When fired, the signal is passed to connected neurons.	Each neuron is a function that operates on an input value (***x***) and a *weight* (***w***). The function is wrapped in an *activation* function that determines whether to pass the output on.



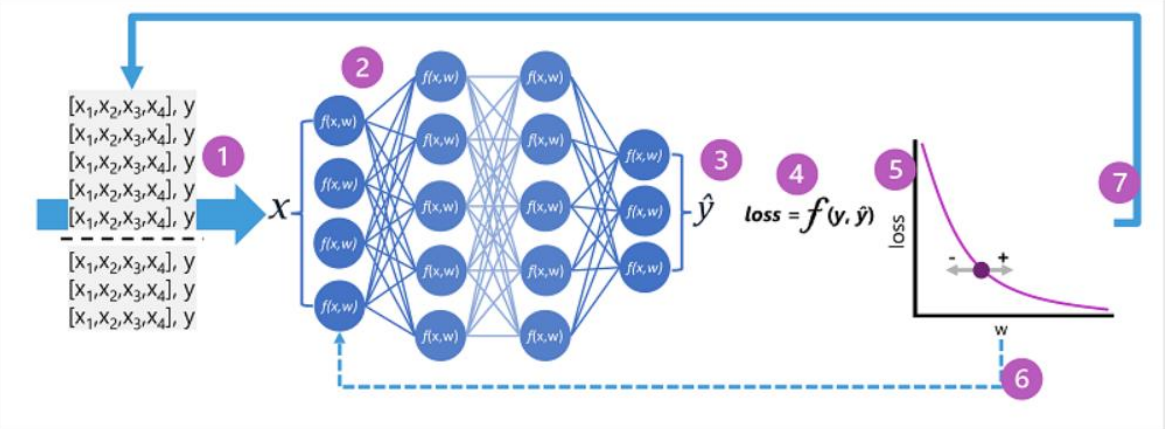
The feature data (*x*) consists of some measurements of a penguin. Specifically, the measurements are:

- The length of the penguin's bill.
- The depth of the penguin's bill.
- The length of the penguin's flippers.
- The penguin's weight.

In this case, ***x*** is a vector of four values, or mathematically, ***x=[x1,x2,x3,x4]***.

The label we're trying to predict (*y*) is the species of the penguin, and that there are three possible species it could be:

- Adelie
- Gentoo
- Chinstrap



900 XP

Fundamentals of Azure AI services

36 min • Module • 8 Units

Feedback

Beginner | AI Engineer | Data Scientist | Student | Azure

In this module, you learn the fundamentals of how Azure AI services can be used to build applications.

Learning objectives

- Understand applications Azure AI services can be used to build
- Understand how to access Azure AI services in the Azure portal
- Understand how to use Azure AI services keys and endpoint for authentication
- Create and use an Azure AI services resource in a Content Safety Studio setting

Introduction

100 XP

2 minutes

Artificial Intelligence (AI) is changing our world and there's hardly an industry that hasn't been affected. From better healthcare to online safety, AI is helping us to tackle some of society's biggest issues.

Azure AI services are a portfolio of AI capabilities that unlock automation for workloads in language, vision, intelligent search, content generation, and much more. They are straightforward to implement and don't require specialist AI knowledge.

Organizations are using Azure AI services in innovative ways, such as within robots to provide life-like companionship to older people by expressing happiness, concern, and even laughter. In other use cases, scientists are using AI to protect endangered species by identifying hard-to-find animals in images. This was previously time-consuming and error-prone work, which the Azure AI Vision service can complete quickly and with a high degree of accuracy, freeing scientists to do other work.

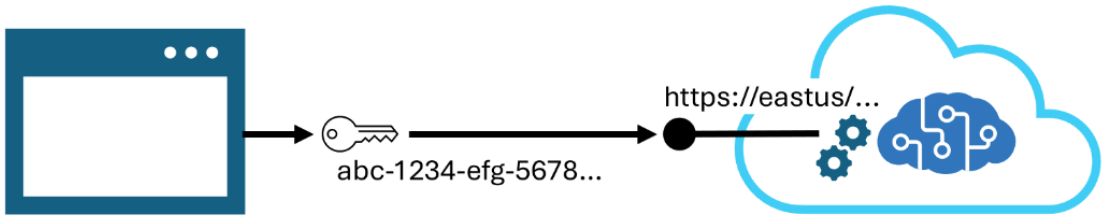
In this module you will learn what Azure AI services are, and how you can use them in your own applications.

Azure AI services are based on three principles that dramatically improve speed-to-market:

- Prebuilt and ready to use
- Accessed through APIs
- Available on Azure

Azure AI services are accessed through APIs

Azure AI services are designed to be used in different development environments, with minimal coding. Developers can access AI services through REST APIs, client libraries, or integrate them with tools such as Logic Apps and Power Automate. APIs are application programming interfaces that define the information that is required for one component to use the services of the other. APIs enable software components to communicate, so one side can be updated without stopping the other from working. Find out more about development options for Azure AI services here.



Azure AI services are available on Azure

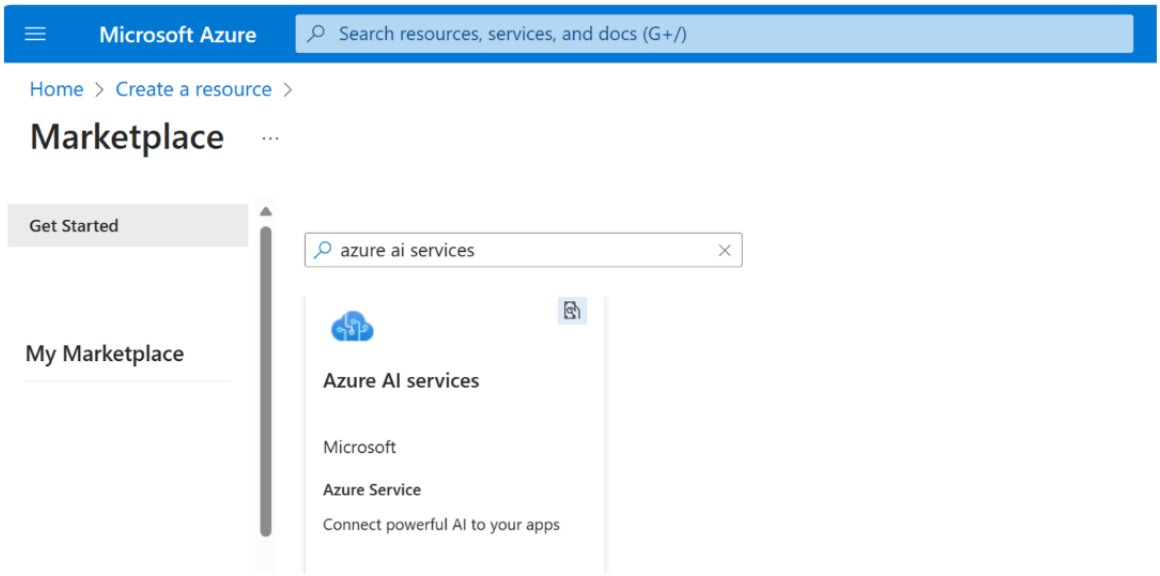
AI services are cloud-based and accessed through Azure resource. This means that they're managed in the same way as other Azure services, such as platform as a service (PaaS), infrastructure as a service (IaaS), or a managed database service. The Azure platform and Resource Manager provide a consistent framework for all your Azure services, from creating or deleting resources, to availability and billing.

Azure AI services are cloud-based, and like all Azure services you need to create a resource to use them. There are two types of AI service resources: multi-service or single-service. Your development requirements and how you want costs to be billed determine the types of resources you need.

- **Multi-service resource**: a resource created in the Azure portal that provides access to multiple Azure AI services with a single key and endpoint. Use the resource **Azure AI services** when you need several AI services or are exploring AI capabilities. When you use an Azure AI services resource, all your AI services are billed together.
- **Single-service resources**: a resource created in the Azure portal that provides access to a single Azure AI service, such as Speech, Vision, Language, etc. Each Azure AI service has a unique key and endpoint. These resources might be used when you only require one AI service or want to see cost information separately.

How to use the Azure portal to create an Azure AI services resource

To create an Azure AI services resource, sign in to the Azure portal with Contributor access and select **Create a resource**. To create a multi-services resource search for *Azure AI services* in the marketplace.



Use Azure AI services

100 XP

3 minutes

Once you create an Azure AI service resource, you can build applications using the REST API, software development kits (SDKs), or visual studio interfaces.

700 XP

Fundamentals of Facial Recognition

27 min • Module • 6 Units

Feedback

Beginner | AI Engineer | Data Scientist | Developer | Solution Architect | Student | Azure AI services

Face detection, analysis, and recognition are important capabilities for artificial intelligence (AI) solutions. Azure AI Face service in Azure makes it easy integrate these capabilities into your applications.

Learning objectives

Learn how to use Azure AI Face service to detect and analyze faces in images.

Uses of face detection and analysis

There are many applications for face detection, analysis, and recognition. For example,

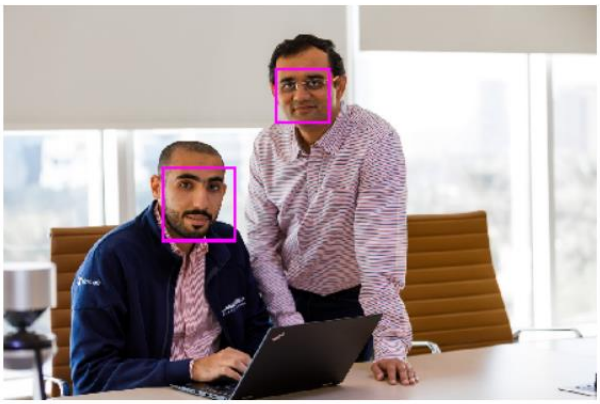
- Security - facial recognition can be used in building security applications, and increasingly it is used in smart phones operating systems for unlocking devices.
- Social media - facial recognition can be used to automatically tag known friends in photographs.
- Intelligent monitoring - for example, an automobile might include a system that monitors the driver's face to determine if the driver is looking at the road, looking at a mobile device, or shows signs of tiredness.
- Advertising - analyzing faces in an image can help direct advertisements to an appropriate demographic audience.
- Missing persons - using public cameras systems, facial recognition can be used to identify if a missing person is in the image frame.
- Identity validation - useful at ports of entry kiosks where a person holds a special entry permit.

Understand Face analysis

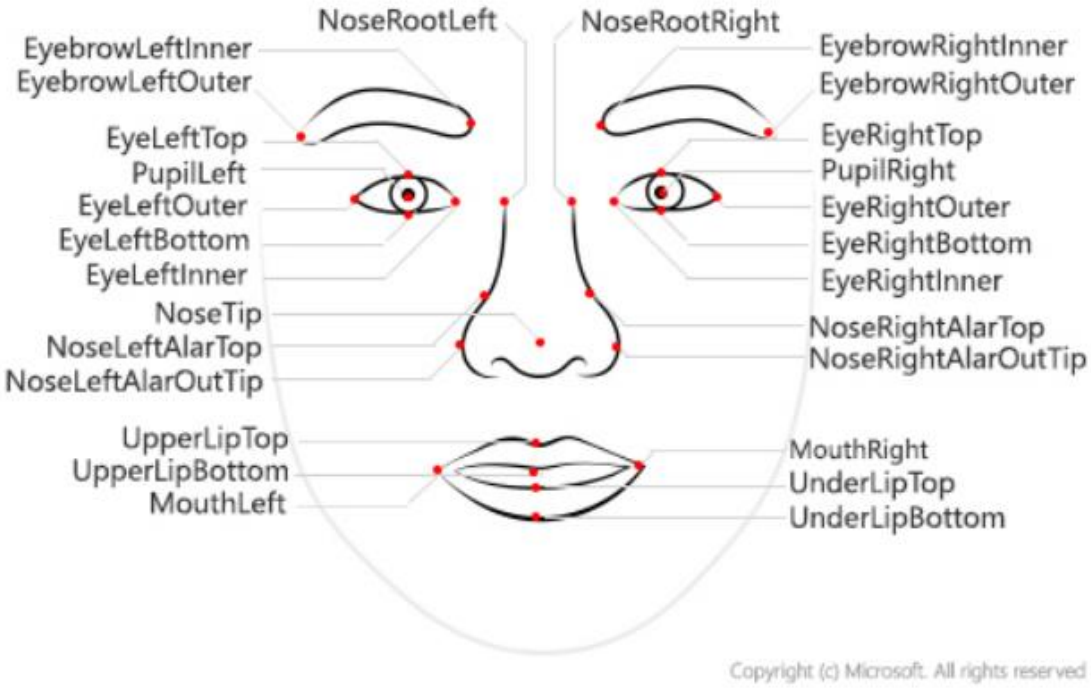
100 XP

3 minutes

Face detection involves identifying regions of an image that contain a human face, typically by returning *bounding box* coordinates that form a rectangle around the face, like this:



With **Face analysis**, facial features can be used to train machine learning models to return other information, such as facial features such as nose, eyes, eyebrows, lips, and others.



Facial recognition

A further application of facial analysis is to train a machine learning model to identify known individuals from their facial features. This is known as *facial recognition*, and uses multiple images of an individual to train the model. This trains the model so that it can detect those individuals in new images on which it wasn't trained.

When used responsibly, facial recognition is an important and useful technology that can improve efficiency, security, and customer experiences. Next we'll explore **Azure AI Face** service, which provides pre-trained models to detect, recognize, and analyze faces.

Get started with Face analysis on Azure

100 XP

4 minutes

Microsoft Azure provides multiple Azure AI services that you can use to detect and analyze faces, including:

- **Azure AI Vision**, which offers face detection and some basic face analysis, such as returning the bounding box coordinates around an image.
- **Azure AI Video Indexer**, which you can use to detect and identify faces in a video.
- **Azure AI Face**, which offers pre-built algorithms that can detect, recognize, and analyze faces.

Of these, Face offers the widest range of facial analysis capabilities.

Face service

The Azure Face service can return the rectangle coordinates for any human faces that are found in an image, as well as a series of attributes related to those face such as:

- **Accessories**: indicates whether the given face has accessories. This attribute returns possible accessories including headwear, glasses, and mask, with confidence score between zero and one for each accessory.
- **Blur**: how blurred the face is, which can be an indication of how likely the face is to be the main focus of the image.
- **Exposure**: such as whether the image is underexposed or over exposed. This applies to the face in the image and not the overall image exposure.
- **Glasses**: whether or not the person is wearing glasses.
- **Head pose**: the face's orientation in a 3D space.
- **Mask**: indicates whether the face is wearing a mask.
- **Noise**: refers to visual noise in the image. If you have taken a photo with a high ISO setting for darker settings, you would notice this noise in the image. The image looks grainy or full of tiny dots that make the image less clear.
- **Occlusion**: determines if there might be objects blocking the face in the image.

Azure resources for Face

To use the Face service, you must create one of the following types of resource in your Azure subscription:

- **Face**: Use this specific resource type if you don't intend to use any other Azure AI services, or if you want to track utilization and costs for Face separately.
- **Azure AI services**: A general resource that includes Azure AI Face along with many other Azure AI services such as Azure AI Content Safety, Azure AI Language, and others. Use this resource type if you plan to use multiple Azure AI services and want to simplify administration and development.

Tips for more accurate results

There are some considerations that can help improve the accuracy of the detection in the images:

- Image format - supported images are JPEG, PNG, GIF, and BMP.
- File size - 6 MB or smaller.
- Face size range - from 36 x 36 pixels up to 4096 x 4096 pixels. Smaller or larger faces will not be detected.
- Other issues - face detection can be impaired by extreme face angles, extreme lighting, and occlusion (objects blocking the face such as a hand).

700 XP

Fundamentals of optical character recognition

27 min • Module • 6 Units

Feedback

Beginner | AI Engineer | Data Scientist | Developer | Solution Architect | Student | Azure AI services

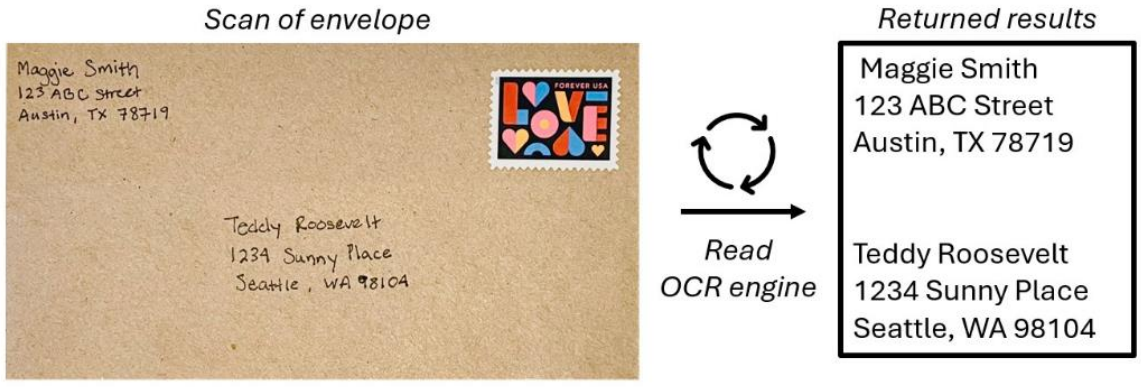
Optical character recognition (OCR) enables artificial intelligence (AI) systems to read text in images, enabling applications to extract information from photographs, scanned documents, and other sources of digitized text.

Learning objectives

Learn how to read text in images with Azure AI Vision.

Uses of OCR

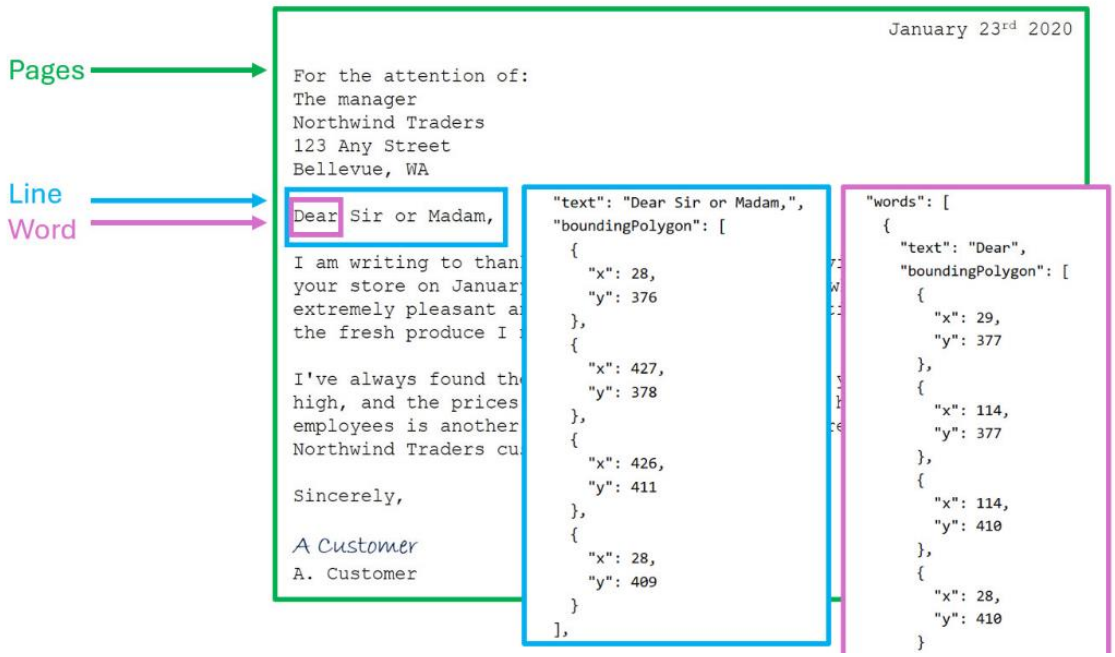
Automating text processing can improve the speed and efficiency of work by removing the need for manual data entry. The ability to recognize printed and handwritten text in images is beneficial in scenarios such as note taking, digitizing medical records or historical documents, scanning checks for bank deposits, and more.



Calling the Read API returns results arranged into the following hierarchy:

- **Pages** - One for each page of text, including information about the page size and orientation.
- **Lines** - The lines of text on a page.
- **Words** - The words in a line of text, including the bounding box coordinates and text itself.

Each line and word includes bounding box coordinates indicating its position on the page.



Get started with Vision Studio on Azure

100 XP

3 minutes

To use the Azure AI Vision service you must first create a resource for it in your Azure subscription. You can use either of the following resource types:

- **Azure AI Vision**: A specific resource for vision services. Use this resource type if you don't intend to use any other AI services, or if you want to track utilization and costs for your AI Vision resource separately.
- **Azure AI services**: A general resource that includes Azure AI Vision along with many other Azure AI services such as Azure AI Language, Azure AI Speech, and others. Use this resource type if you plan to use multiple Azure AI services and want to simplify administration and development.

Once you've created a resource, there are several ways to use Azure AI Vision's Read API:

- Vision Studio
- REST API
- Software Development Kits (SDKs): Python, C#, JavaScript

700 XP

Fundamentals of Text Analysis with the Language Service

41 min • Module • 6 Units

Feedback

Beginner | AI Engineer | Data Scientist | Developer | Solution Architect | Student | Azure AI services

Explore Azure AI Language's natural language processing (NLP) features, which include sentiment analysis, key phrase extraction, named entity recognition, and language detection.

Learning objectives

Learn how to use Azure AI Language for text analysis

Introduction

100 XP

3 minutes

In order for computer systems to interpret the subject of a text in a similar way humans do, they use **natural language processing** (NLP), an area within AI that deals with understanding written or spoken language, and responding in kind. *Text analysis* describes NLP processes that extract information from unstructured text.

Natural language processing might be used to create:

- A social media feed analyzer that detects sentiment for a product marketing campaign.
- A document search application that summarizes documents in a catalog.
- An application that extracts brands and company names from text.

Azure AI Language is a cloud-based service that includes features for understanding and analyzing text. Azure AI Language includes various features that support sentiment analysis, key phrase identification, text summarization, and conversational language understanding.

In this module, you'll explore the capabilities of text analytics, and how you might use them.

Understand Text Analytics

100 XP

5 minutes

Before exploring the text analytics capabilities of the Azure AI Language service, let's examine some general principles and common techniques used to perform text analysis and other natural language processing (NLP) tasks.

Some of the earliest techniques used to analyze text with computers involve statistical analysis of a body of text (a *corpus*) to infer some kind of semantic meaning. Put simply, if you can determine the most commonly used words in a given document, you can often get a good idea of what the document is about.

Tokenization

The first step in analyzing a corpus is to break it down into *tokens*. For the sake of simplicity, you can think of each distinct word in the training text as a token, though in reality, tokens can be generated for partial words, or combinations of words and punctuation.

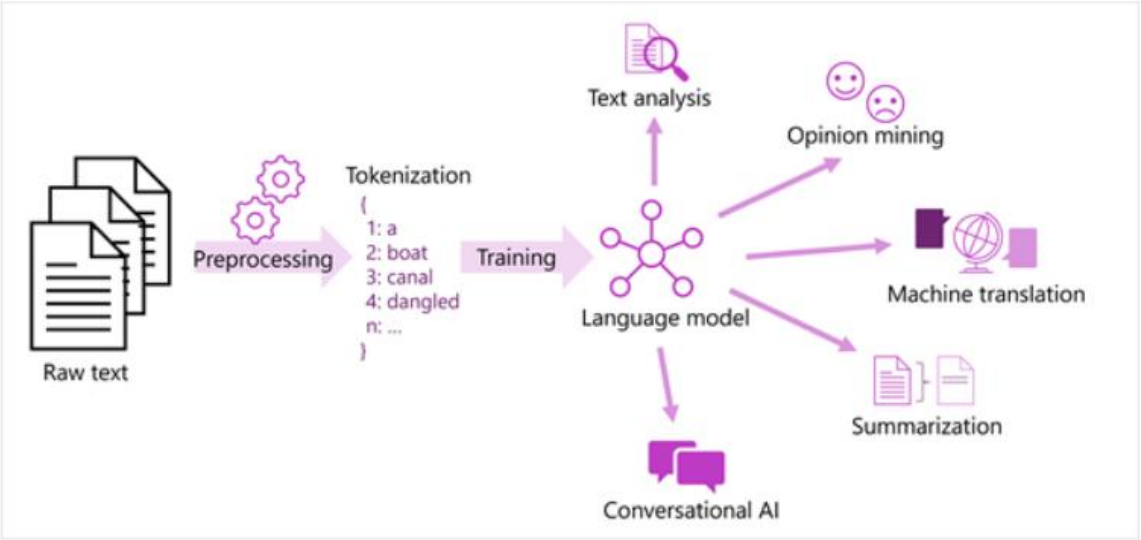
For example, consider this phrase from a famous US presidential speech: "*we choose to go to the moon*". The phrase can be broken down into the following tokens, with numeric identifiers:

1. we
2. choose
3. to
4. go
5. the
6. moon

Notice that "to" (token number 3) is used twice in the corpus. The phrase "*we choose to go to the moon*" can be represented by the tokens [*1,2,3,4,3,5,6*].

Frequency analysis

After tokenizing the words, you can perform some analysis to count the number of occurrences of each token. The most commonly used words (other than *stop words* such as "*a*", "*the*", and so on) can often provide a clue as to the main subject of a text corpus. For example, the most common words in the entire text of the "go to the moon" speech we considered previously include "*new*", "*go*", "*space*", and "*moon*". If we were to tokenize the text as bi-grams (word pairs), the most common bi-gram in the speech is "*the moon*". From this information, we can easily surmise that the text is primarily concerned with space travel and going to the moon.



Get started with text analysis

100 XP

10 minutes

Azure AI Language is a part of the Azure AI services offerings that can perform advanced natural language processing over unstructured text. Azure AI Language's text analysis features include:

- **Named entity recognition** identifies people, places, events, and more. This feature can also be customized to extract custom categories.
- **Entity linking** identifies known entities together with a link to Wikipedia.
- **Personal identifying information (PII) detection** identifies personally sensitive information, including personal health information (PHI).
- **Language detection** identifies the language of the text and returns a language code such as "en" for English.
- **Sentiment analysis and opinion mining** identifies whether text is positive or negative.
- **Summarization** summarizes text by identifying the most important information.
- **Key phrase extraction** lists the main concepts from unstructured text.

700 XP

Fundamentals of question answering with the Language Service

29 min • Module • 6 Units

Feedback

Beginner | AI Engineer | Data Scientist | Developer | Solution Architect | Student | Azure AI Bot Service

Create a custom question answering knowledge base with Azure AI Language and create a bot with Azure AI Bot Service that answers user questions.

Learning objectives

After completing this module, you'll be able to understand how to use Azure AI Language and Azure AI Bot Service to create a bot.

Introduction

100 XP

1 minute

We are used to being able to communicate at any time of the day or night, anywhere in the world, putting organizations under pressure to react fast enough to their customers. We want personal responses to our queries, without having to read in-depth documentation to find answers. This often means that support staff get overloaded with requests for help through multiple channels, and that people are left waiting for a response.

Conversational AI describes solutions that enable a dialog between an AI agent and a human. Generically, conversational AI agents are known as bots. People can engage with bots through channels such as web chat interfaces, email, social media platforms, and more.

Azure AI Language's question answering feature provides you with the ability to create conversational AI solutions. Next you'll learn about question answering.

Get started with the Language service and Azure Bot Service

100 XP

3 minutes

You can easily create a user support bot solution on Microsoft Azure using a combination of two core services:

- **Azure AI Language**: includes a custom question answering feature that enables you to create a knowledge base of question and answer pairs that can be queried using natural language input.
- **Azure AI Bot Service**: provides a framework for developing, publishing, and managing bots on Azure.

Define questions and answers

After provisioning a Language resource, you can use the Language Studio's custom question answering feature to create a project that consists of question-and-answer pairs. These questions and answers can be:

- Generated from an existing FAQ document or web page.
- Entered and edited manually.

In many cases, a project is created using a combination of all of these techniques; starting with a base dataset of questions and answers from an existing FAQ document and extending the knowledge base with additional manual entries.

Questions in the project can be assigned *alternative phrasing* to help consolidate questions with the same meaning. For example, you might include a question like:

> *What is your head office location?*

You can anticipate different ways this question could be asked by adding an alternative phrasing such as:

> *Where is your head office located?*

700 XP

Fundamentals of conversational language understanding

42 min • Module • 6 Units

Feedback

Intermediate | AI Engineer | Developer | Solution Architect | Student | Azure | Azure AI Bot Service | Azure SDKs

In this module, we introduce you to conversational language understanding, and show how to create applications that understand language with Azure AI Language.

Learning objectives

In this module, you'll:

- Learn what conversational language understanding is.
- Learn about key features, such as intents and utterances.
- Build and publish a natural-language machine-learning model.

Describe conversational language understanding

100 XP

3 minutes

To work with conversational language understanding, you need to take into account three core concepts: *utterances*, *entities*, and *intents*.

Utterances

An utterance is an example of something a user might say, and which your application must interpret. For example, when using a home automation system, a user might use the following utterances:

> "*Switch the fan on.*"
>
> "*Turn on the light.*"

Entities

An entity is an item to which an utterance refers. For example, **fan** and **light** in the following utterances:

> "*Switch the **fan** on.*"
>
> "*Turn on the **light**.*"

Intents

An intent represents the purpose, or goal, expressed in a user's utterance. For example, for both of the previously considered utterances, the intent is to turn a device on; so in your conversational language understanding application, you might define a **TurnOn** intent that is related to these utterances.

A conversational language understanding application defines a model consisting of intents and entities. Utterances are used to train the model to identify the most likely intent and the entities to which it should be applied based on a given input. The home assistant application we've been considering might include multiple intents, like the following examples:

Expand table

Intent	Related Utterances	Entities
Greeting	"Hello"	
	"Hi"	
	"Hey"	
	"Good morning"	
TurnOn	"Switch the fan on"	fan (device)
	"Turn the light on"	light (device)

Authoring

After you've created an authoring resource, you can use it to train a conversational language understanding model. To train a model, start by defining the entities and intents that your application will predict as well as utterances for each intent that can be used to train the predictive model.

Conversational language understanding provides a comprehensive collection of prebuilt *domains* that include pre-defined intents and entities for common scenarios; which you can use as a starting point for your model. You can also create your own entities and intents.

When you create entities and intents, you can do so in any order. You can create an intent, and select words in the sample utterances you define for it to create entities for them; or you can create the entities ahead of time and then map them to words in utterances as you're creating the intents.

You can write code to define the elements of your model, but in most cases it's easiest to author your model using the Language studio - a web-based interface for creating and managing Conversational Language Understanding applications.

700 XP

Fundamentals of Azure AI Speech

30 min • Module • 6 Units

Feedback

Beginner | AI Engineer | Student | Azure

Learn how to recognize and synthesize speech by using Azure AI Speech.

Learning objectives

In this module you will:

- Learn about speech recognition and synthesis
- Learn how to use Azure AI Speech

Introduction

100 XP

1 minute

AI speech capabilities enable us to manage home and auto systems with voice instructions, get answers from computers for spoken questions, generate captions from audio, and much more.

To enable this kind of interaction, the AI system must support two capabilities:

- **Speech recognition** - the ability to detect and interpret spoken input
- **Speech synthesis** - the ability to generate spoken output

Azure AI Speech provides speech to text and text to speech capabilities through speech recognition and synthesis. You can use prebuilt and custom Speech service models for a variety of tasks, from transcribing audio to text with high accuracy, to identifying speakers in conversations, creating custom voices, and more. Next you'll learn how AI speech capabilities work.

Understand speech recognition and synthesis

100 XP

3 minutes

Speech recognition takes the spoken word and converts it into data that can be processed - often by transcribing it into text. The spoken words can be in the form of a recorded voice in an audio file, or live audio from a microphone. Speech patterns are analyzed in the audio to determine recognizable patterns that are mapped to words. To accomplish this, the software typically uses multiple models, including:

- An *acoustic* model that converts the audio signal into phonemes (representations of specific sounds).
- A *language* model that maps phonemes to words, usually using a statistical algorithm that predicts the most probable sequence of words based on the phonemes.

The recognized words are typically converted to text, which you can use for various purposes, such as:

- Providing closed captions for recorded or live videos
- Creating a transcript of a phone call or meeting
- Automated note dictation
- Determining intended user input for further processing

Speech synthesis is concerned with vocalizing data, usually by converting text to speech. A speech synthesis solution typically requires the following information:

- The text to be spoken
- The voice to be used to vocalize the speech

To synthesize speech, the system typically *tokenizes* the text to break it down into individual words, and assigns phonetic sounds to each word. It then breaks the phonetic transcription into *prosodic* units (such as phrases, clauses, or sentences) to create phonemes that will be converted to audio format. These phonemes are then synthesized as audio and can be assigned a particular voice, speaking rate, pitch, and volume.

You can use the output of speech synthesis for many purposes, including:

- Generating spoken responses to user input
- Creating voice menus for telephone systems
- Reading email or text messages aloud in hands-free scenarios
- Broadcasting announcements in public locations, such as railway stations or airports

Get started with speech on Azure

100 XP

3 minutes

Microsoft Azure offers both speech recognition and speech synthesis capabilities through **Azure AI Speech** service, which includes the following application programming interfaces (APIs):

- The **Speech to text** API
- The **Text to speech** API

700 XP

Fundamentals of Azure AI Document Intelligence

26 min • Module • 6 Units

Feedback

Beginner | AI Engineer | Data Scientist | Developer | Solution Architect | Student | Azure AI services

Document processing is a common task in many business scenarios. Organizations can use Azure AI Document Intelligence to automate data extraction across document types, such as receipts, invoices, and more.

Learning objectives

Learn how to use the prebuilt receipt processing capabilities of Azure AI Document Intelligence.

Azure AI Document Intelligence supports features that can analyze documents and forms with prebuilt and custom models. In this module, you explore how Azure AI services provide access to document intelligence capabilities.

Get started with receipt analysis on Azure

100 XP

3 minutes

Azure AI Document Intelligence consists of features grouped by model type:

- **Prebuilt models** - pretrained models that have been built to process common document types such as invoices, business cards, ID documents, and more. These models are designed to recognize and extract specific fields that are important for each document type.
- **Custom models** - can be trained to identify specific fields that are not included in the existing pretrained models.
- **Document analysis** - general document analysis that returns structured data representations, including regions of interest and their inter-relationships.

Prebuilt models

The prebuilt models apply advanced machine learning to accurately identify and extract text, key-value pairs, tables, and structures from forms and documents. These capabilities include extracting:

- customer and vendor details from invoices
- sales and transaction details from receipts
- identification and verification details from identity documents
- health insurance details
- business contact details
- agreement and party details from contracts
- taxable compensation, mortgage interest, student loan details and more

For example, consider the prebuilt receipt model. It processes receipts by:

- Matching field names to values
- Identifying tables of data
- Identifying specific fields, such as dates, telephone numbers, addresses, totals, and others

The receipt model has been trained to recognize data on several different receipt types, such as thermal receipts (printed on heat-sensitive paper), hotel receipts, gas receipts, credit card receipts, and parking receipts.

Fields recognized include:

- Name, address, and telephone number of the merchant
- Date and time of the purchase
- Name, quantity, and price of each item purchased
- Total, subtotals, and tax values

Each field and data pair has a confidence level, indicating the likely level of accuracy. This could be used to automatical identify when a person needs to verify a receipt.

The model has been trained to recognize several different languages, depending on the receipt type. For best results when using the prebuilt receipt model, images should be:

- JPEG, PNG, BMP, PDF, or TIFF format
- File size less than 500 MB for paid (S0) tier and 4 MB for free (F0) tier
- Between 50 x 50 pixels and 10000 x 10000 pixels
- For PDF documents, no larger than 17 inches x 17 inches
- One receipt per document

You can get started with training models in the Document Intelligence Studio, a user interface for testing document analysis, prebuilt models, and creating custom models.

1300 XP

Fundamentals of Knowledge Mining and Azure AI Search

1 hr 5 min • Module • 12 Units

Feedback

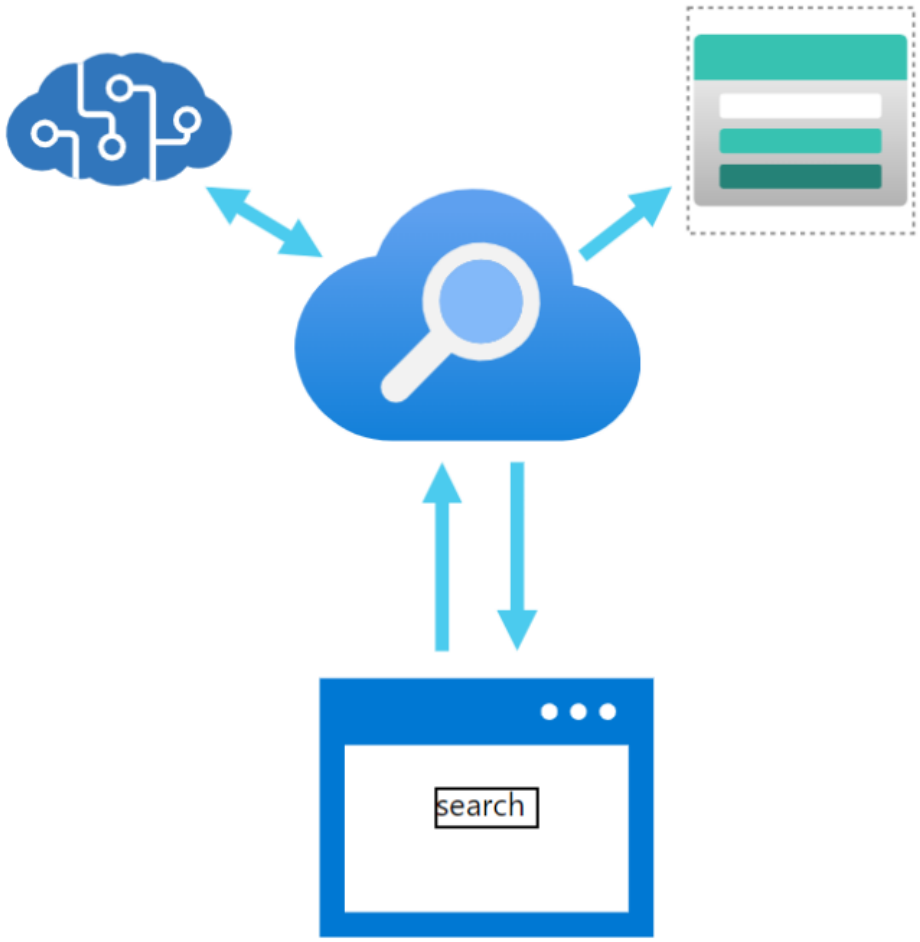
Beginner | Developer | Azure | Azure AI Search

Use Azure AI Search to make your data searchable.

Learning objectives

In this module, you will:

- Explore Azure AI Search
- Create an Azure AI Search index
- Import data to the index
- Query the Azure AI Search index



Azure AI Search exists to complement existing technologies and provides a programmable search engine built on Apache Lucene, an open-source software library. It's a highly available platform offering a 99.9% uptime SLA available for cloud and on-premises assets.

Azure AI Search comes with the following features:

- **Data from any source**: accepts data from any source provided in JSON format, with auto crawling support for selected data sources in Azure.
- **Full text search and analysis**: offers full text search capabilities supporting both simple query and full Lucene query syntax.
- **AI powered search**: has Azure AI capabilities built in for image and text analysis from raw content.
- **Multi-lingual** offers linguistic analysis for 56 languages to intelligently handle phonetic matching or language-specific linguistics. Natural language processors available in Azure AI Search are also used by Bing and Office.
- **Geo-enabled**: supports geo-search filtering based on proximity to a physical location.
- **Configurable user experience**: has several features to improve the user experience including autocomplete, autosuggest, pagination, and hit highlighting.

Built in skills

Built-in skills are based on pretrained models from Microsoft, which means you can't train the model using your own training data. Skills that call the Azure AI services APIs have a dependency on those services and are billed at the Azure AI services pay-as-you-go price when you attach a resource. Other skills are metered by Azure AI Search, or are utility skills that are available at no charge.

Built-in skills fall into these categories:

Natural language processing skills: with these skills, unstructured text is mapped as searchable and filterable fields in an index.

Some examples include:

- Key Phrase Extraction: uses a pre-trained model to detect important phrases based on term placement, linguistic rules, proximity to other terms, and how unusual the term is within the source data.
- Text Translation Skill: uses a pre-trained model to translate the input text into various languages for normalization or localization use cases.

Image processing skills: creates text representations of image content, making it searchable using the query capabilities of Azure AI Search.

Some examples include:

- Image Analysis Skill: uses an image detection algorithm to identify the content of an image and generate a text description.
- Optical Character Recognition Skill: allows you to extract printed or handwritten text from images, such as photos of street signs and products, as well as from documents—invoices, bills, financial reports, articles, and more.

Create an index in the Azure portal

100 XP

3 minutes

Before using an indexer to create an index, you'll first need to make your data available in a supported data source. Supported data sources include:

- Cosmos DB (SQL API)
- Azure SQL (database, managed instance, and SQL Server on an Azure VM)
- Azure Storage (Blob Storage, Table Storage, ADLS Gen2)

Using the Azure portal's Import data wizard

Once your data is in an Azure data source, you can begin using Azure AI Search. Contained within the Azure AI Search service in Azure portal is the Import data wizard, which automates processes in the Azure portal to create various objects needed for the search engine. You can see it in action when creating any of the following objects using the Azure portal:

- **Data Source**: Persists connection information to source data, including credentials. A data source object is used exclusively with indexers.
- **Index**: Physical data structure used for full text search and other queries.
- **Indexer**: A configuration object specifying a data source, target index, an optional AI skillset, optional schedule, and optional configuration settings for error handling and base-64 encoding.
- **Skillset**: A complete set of instructions for manipulating, transforming, and shaping content, including analyzing and extracting information from image files. Except for very simple and limited structures, it includes a reference to an Azure AI services resource that provides enrichment.
- **Knowledge store**: Stores output from an AI enrichment pipeline in tables and blobs in Azure Storage for independent analysis or downstream processing.

1200 XP

Fundamentals of Generative AI

1 hr 12 min remaining • Module • 1 of 11 units completed

Feedback

Beginner | AI Engineer | Developer | Solution Architect | Student | Azure OpenAI Service | Azure

In this module, you explore the way in which language models enable AI applications and services to generate original content based on natural language input. You also learn how generative AI enables the creation of copilots that can assist humans in creative tasks.

Learning objectives

By the end of this module, you are able to:

- Understand generative AI's place in the development of artificial intelligence.
- Understand language models and their role in intelligent applications.
- Describe examples of copilots and good prompts.

Introduction

100 XP

1 minute

Generative AI, and technologies that implement it like Microsoft Copilot are increasingly in the public consciousness – even among people who don't work in technology roles or have a background in computer science or machine learning. The futurist and novelist Arthur C. Clarke is quoted as observing that "any sufficiently advanced technology is indistinguishable from magic." In the case of generative AI there does indeed seem to be an almost miraculous ability to produce human-like original content, including poetry, prose, and even computer code.

However, there's no wizardry involved in generative AI – just the application of mathematical techniques incrementally discovered and refined over many years of research into statistics, data science, and machine learning. You can gain a high-level understanding of how the magic trick is done by learning the core concepts and principles explored in this module. As you learn more about the generative AI technologies we have today, you can help society imagine new possibilities for AI tomorrow.

What is generative AI?

100 XP

2 minutes

Artificial Intelligence (AI) imitates human behavior by using machine learning to interact with the environment and execute tasks without explicit directions on what to output.

Generative AI describes a category of capabilities within AI that create original content. People typically interact with generative AI that has been built into chat applications. One popular example of such an application is Microsoft Copilot, a chatbot companion to browse the web more effectively.

Generative AI applications take in natural language input, and return appropriate responses in a variety of formats such as natural language, images, or code.

What are language models?

100 XP

8 minutes

Generative AI applications are powered by *language models*, which are a specialized type of machine learning model that you can use to perform *natural language processing* (NLP) tasks, including:

- Determining *sentiment* or otherwise classifying natural language text.
- Summarizing text.
- Comparing multiple text sources for semantic similarity.
- Generating new natural language.

While the mathematical principles behind these language models can be complex, a basic understanding of the architecture used to implement them can help you gain a conceptual understanding of how they work.

Large and small language models

There are many language models available that you can use to power generative AI applications. In general, language models can be considered in two categorize: *Large Language Models* (LLMs) and *Small Language models* (SLMs).

Large Language Models (LLMs)	**Small Language Models (SLMs)**
LLMs are trained with vast quantities of text that represents a wide range of general subject matter – typically by sourcing data from the Internet and other generally available publications.	SLMs are trained with smaller, more subject-focused datasets
When trained, LLMs have many billions (even trillions) of parameters (weights that can be applied to vector embeddings to calculate predicted token sequences).	Typically have fewer parameters than LLMs.
Able to exhibit comprehensive language generation capabilities in a wide range of conversational contexts.	This focused vocabulary makes them very effective in specific conversational topics, but less effective at more general language generation.
Their large size can impact their performance and make them difficult to deploy locally on devices and computers.	The smaller size of SLMs can provide more options for deployment, including local deployment to devices and on-premises computers; and makes them faster and easier to fine-tune.
Fine-tuning the model with additional data to customize its subject expertise can be time-consuming, and expensive in terms of the compute power required to perform the additional training.	Fine-tuning can potentially be less time-consuming and expensive.

What are copilots?

100 XP

3 minutes

The availability of language models has led to the emergence of new ways to interact with applications and systems through digital *copilots*. Copilots are generative AI assistants that are integrated into applications often as chat interfaces. They provide contextualized support for common tasks in those applications.

Microsoft Copilot is integrated into a wide range of Microsoft applications and user experiences. It is based on an open architecture that enables third-party developers to create their own plug-ins to extend or customize the user experience with Microsoft Copilot. Additionally, third-party developers can create their own copilots using the same open architecture.

Business users can use copilots to boost their productivity and creativity with AI-generated content and automation of tasks. Developers can extend copilots by creating plug-ins that integrate them into business processes and data, or even create custom copilots to build generative AI capabilities into apps and services.

Copilots have the potential to revolutionize the way we work by helping with first drafts, information synthesis, strategic planning, and much more. The goal of copilot features is to empower people to be smarter, more productive, more creative, and connected to the people and things around them.

Copilot Studio

Copilot Studio is designed to work well for low-code development scenarios in which technically proficient business users or developers can create conversational AI experiences. The resulting copilot is a fully managed SaaS (software as a service) solution, hosted in your Microsoft 365 environment and delivered through chat channels like Microsoft Teams. With Copilot Studio, the infrastructure considerations and model deployment details are taken care of for you, making it easy to focus on creating an effective solution. For more information, see https://www.microsoft.com/microsoft-copilot/microsoft-copilot-studio.

1200 XP

Fundamentals of Azure OpenAI Service

1 hr 3 min • Module • 11 Units

Feedback

Beginner | AI Engineer | Data Scientist | Student | Azure AI Bot Service | Azure AI services | Azure Machine Learning

Get to know the connection between artificial intelligence (AI), Responsible AI, and text, code, and image generation. Understand how you can use Azure OpenAI to build solutions against AI models within Azure.

Learning objectives

In this module you'll learn how to:

- Describe Azure OpenAI workloads and access the Azure OpenAI Service
- Understand generative AI models
- Understand Azure OpenAI's language, code, and image capabilities
- Understand Azure OpenAI's responsible AI practices and limited access policies

Capabilities of OpenAI AI models

There are several categories of capabilities found in OpenAI AI models, three of these include:

Expand table

Capability	Examples
Generating natural language	Such as: summarizing complex text for different reading levels, suggesting alternative wording for sentences, *and much more*
Generating code	Such as: translating code from one programming language into another, identifying and troubleshooting bugs in code, *and much more*
Generating images	Such as: generating images for publications from text descriptions *and much more*

Introduction to Azure OpenAI Service

Azure OpenAI Service is a result of the partnership between Microsoft and OpenAI. The service combines Azure's enterprise-grade capabilities with OpenAI's generative AI model capabilities.

Azure OpenAI is available for Azure users and consists of four components:

- Pre-trained generative AI models
- Customization capabilities; the ability to fine-tune AI models with your own data
- Built-in tools to detect and mitigate harmful use cases so users can implement AI responsibly
- Enterprise-grade security with role-based access control (RBAC) and private networks

Using Azure OpenAI allows you to transition between your work with Azure services and OpenAI, while utilizing Azure's private networking, regional availability, and responsible AI content filtering.

Understand Azure OpenAI workloads

Common AI workloads include machine learning, computer vision, natural language processing, conversational AI, anomaly detection, and knowledge mining.

Azure OpenAI supports many **generative** AI workloads such as:

- **Generating Natural Language**
  - *Text completion*: generate and edit text
  - *Embeddings*: search, classify, and compare text
- **Generating Code**: generate, edit, and explain code
- **Generating Images**: generate and edit images

In the Azure OpenAI Studio, you can build AI models and deploy them for public consumption in software applications. Azure OpenAI's capabilities are made possible by specific generative AI models. Different models are optimized for different tasks; some models excel at summarization and providing general unstructured responses, and others are built to generate code or unique images from text input.

These Azure OpenAI models include:

- **GPT-4** models that represent the latest generative models for natural language and code.
- **GPT-3.5** models that can generate natural language and code responses based on prompts.
- **Embeddings** models that convert text to numeric vectors for analysis - for example comparing sources of text for similarity.
- **DALL-E** models that generate images based on natural language descriptions.

Fundamentals of Responsible Generative AI

1000 XP

50 min remaining • Module • 0 of 9 units completed

Generative AI enables amazing creative solutions, but must be implemented responsibly to minimize the risk of harmful content generation.

Start >

Plan a responsible generative AI solution

100 XP

2 minutes

The Microsoft guidance for responsible generative AI is designed to be practical and actionable. It defines a four stage process to develop and implement a plan for responsible AI when using generative models. The four stages in the process are:

1. *Identify* potential harms that are relevant to your planned solution.
2. *Measure* the presence of these harms in the outputs generated by your solution.
3. *Mitigate* the harms at multiple layers in your solution to minimize their presence and impact, and ensure transparent communication about potential risks to users.
4. *Operate* the solution responsibly by defining and following a deployment and operational readiness plan.

Identify potential harms

100 XP

5 minutes

The first stage in a responsible generative AI process is to identify the potential harms that could affect your planned solution. There are four steps in this stage, as shown here:

1. Identify potential harms
2. Prioritize identified harms
3. Test and verify the prioritized harms
4. Document and share the verified harms

Manual and automatic testing

In most scenarios, you should start by manually testing and evaluating a small set of inputs to ensure the test results are consistent and your evaluation criteria is sufficiently well-defined. Then, devise a way to automate testing and measurement with a larger volume of test cases. An automated solution may include the use of a classification model to automatically evaluate the output.

Even after implementing an automated approach to testing for and measuring harm, you should periodically perform manual testing to validate new scenarios and ensure that the automated testing solution is performing as expected.

1. **Model**
2. **Safety System**
3. **Metaprompt and grounding**
4. **User experience**

1: The *model* layer

The model layer consists of the generative AI model(s) at the heart of your solution. For example, your solution may be built around a model such as GPT-4.

Mitigations you can apply at the model layer include:

- Selecting a model that is appropriate for the intended solution use. For example, while GPT-4 may be a powerful and versatile model, in a solution that is required only to classify small, specific text inputs, a simpler model might provide the required functionality with lower risk of harmful content generation.
- *Fine-tuning* a foundational model with your own training data so that the responses it generates are more likely to be relevant and scoped to your solution scenario.

2: The *safety system* layer

The safety system layer includes platform-level configurations and capabilities that help mitigate harm. For example, Azure OpenAI Service includes support for *content filters* that apply criteria to suppress prompts and responses based on classification of content into four severity levels (*safe*, *low*, *medium*, and *high*) for four categories of potential harm (*hate*, *sexual*, *violence*, and *self-harm*).

Other safety system layer mitigations can include abuse detection algorithms to determine if the solution is being systematically abused (for example through high volumes of automated requests from a bot) and alert notifications that enable a fast response to potential system abuse or harmful behavior.

3: The *metaprompt and grounding* layer

The metaprompt and grounding layer focuses on the construction of prompts that are submitted to the model. Harm mitigation techniques that you can apply at this layer include:

- Specifying *metaprompts* or system inputs that define behavioral parameters for the model.
- Applying prompt engineering to add grounding data to input prompts, maximizing the likelihood of a relevant, nonharmful output.
- Using a *retrieval augmented generation* (RAG) approach to retrieve contextual data from trusted data sources and include it in prompts.

4: The *user experience* layer

The user experience layer includes the software application through which users interact with the generative AI model as well as documentation or other user collateral that describes the use of the solution to its users and stakeholders.

Designing the application user interface to constrain inputs to specific subjects or types, or applying input and output validation can mitigate the risk of potentially harmful responses.

Documentation and other descriptions of a generative AI solution should be appropriately transparent about the capabilities and limitations of the system, the models on which it's based, and any potential harms that may not always be addressed by the mitigation measures you have put in place.

Complete prerelease reviews

Before releasing a generative AI solution, identify the various compliance requirements in your organization and industry and ensure the appropriate teams are given the opportunity to review the system and its documentation. Common compliance reviews include:

- Legal
- Privacy
- Security
- Accessibility